I B.Tech CSE - PPSC

C - Programming Language

INTRODUCTION TO C : C is a general purpose programming language. C compilers are commonly available for computers of all sizes. The compilers are usually compact and they generate object programs that are small and highly efficient, when compared with programs compiled from other high level languages. C programs are highly portable.

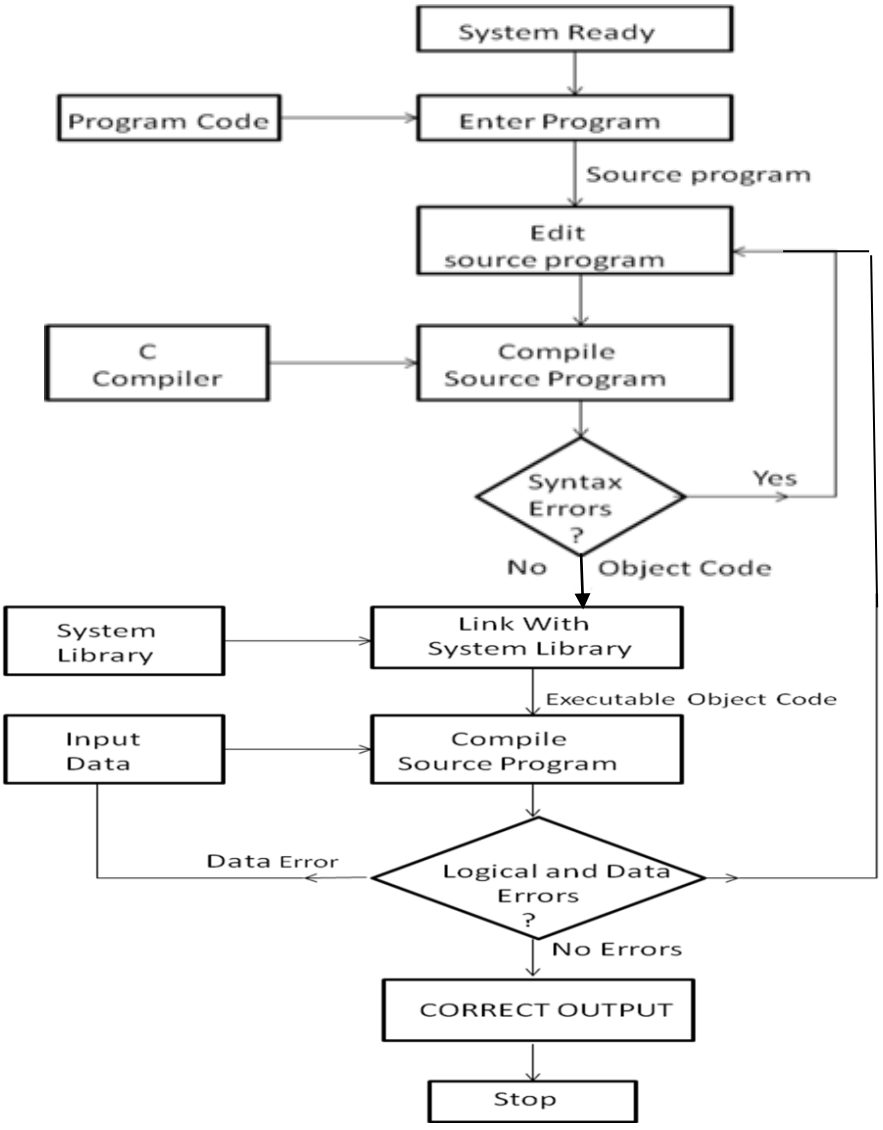
History of C: C was originally developed in the 1970's by Dennis Ritchie at Bell Telephone Laboratories, Inc. It is an outgrowth of two earlier languages, called BCPL and B, which were also developed at Bell Laboratories.

PROGRAM DEVELOPMENT STEPS

Executing a Program written in C involves a series of steps. These are:

1. Creating the program.
2. Compiling the program.
3. Linking the program with functions that are needed from C library.
4. Executing the program.

The Following Figure shows the process of creating, compiling and executing a C Program



Creating a Program:

Once we load the Operating system into the memory, the computer is ready to receive the program. The program must be entered into a *file*. The file name can consist of letters, digits and special characters, followed by a dot and a letter c.

Ex: hello.c
program.c
ex1.c

The file is created with the help of a text editor, we use tc.exe**. double click the** icon TC on the desktop. TURBOC editor will be opened. If the file is existed before, it can be loaded. If it does not exist, the file has to be created. After typing the instructions save it with file extension .c.

When editing is over, the file is saved on disk which can be referred later by its name. the program that is entered into the file is known as the source program.

Compiling the program: to compile the program press ALT + F9

Now the source program instructions are transformed into a form suitable for execution by the computer. If mistakes in syntax or semantics of the language are discovered, they are listed out and the compilation process ends right there. The errors should be corrected in the source program with the help of the editor and the compilation is done again.

Linking the program:

Linking is the process of putting together other program files and functions that are required by the program. The compiled and linked program is called the *executable object code*

Executing the program: to execute the program press CTRL+F9

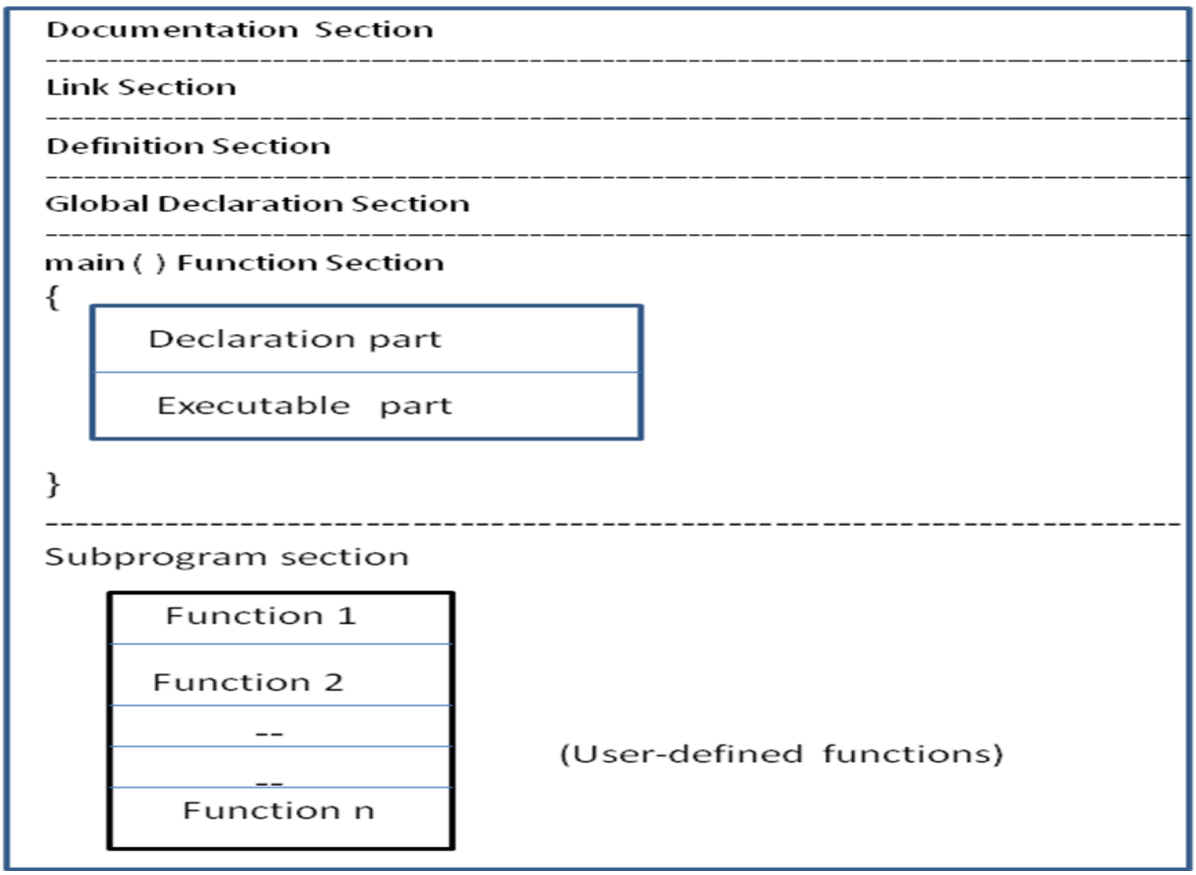
would load the executable object code into the computer memory and execute the instructions.

Some times the program may request for some data to be entered through the keyboard. When there is wrong with the program logic or data, then it is necessary to correct the source program or data.

In case the source program is modified the entire process of compiling, linking and executing the program should be repeated.

STRUCTURE OF A C PROGRAM:

Every C program consists of one or more functions, one of, which must be called main(). The program will always begin by executing the main() function. Additional function definitions may precede or follow main.



Documentation Section: The Documentation Section consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later.

Link Section: The link Section provides instructions to the compiler to link functions from the system library.

Definition section: The definition section defines all symbolic constants.

Global Declaration Section: The Global declaration section declares all the global variables that are used in more than one function. This section also declares all the user-defined functions.

main() function section:

Every C Program must have one **main() function section**. This Section consists of two parts.

1. Declaration part
2. Executable part

The **Declaration part** declares all the variables used in the executable part. There is at least one statement in the **executable part**. These two parts must appear between the opening brace and closing braces.

The program execution begins at the opening brace and ends at the closing brace. The closing brace of the main function is the logical end of the program. All Statements in the declaration and executable parts end with a semicolon (;).

sub program section: The **sub program section** contains all the user-defined functions that are called in the main function. User defined functions are generally placed immediately after the main function, although they may appear in any order.

All functions except the main function may be absent when they are not required.

Sample C Programs:

/*----------Program to add two numbers ----------*/

Algorithm
step 1) start
step 2) read a
step 3) read b
step4) sum=a+b
step 5) display sum
step 6) stop

```
#include <stdio.h>
#include <conio.h>
 void main()
 {
    int a,b,sum;
    clrscr();
    printf("enter a and b : ");
    scanf("%d",&a);
    scanf("%d",&b);
    sum=a+b;
    printf("sum of two numbers = %d\n",sum);
    getch();
 }
```

OUTPUT

```
enter a and b : 10 20
sum of two numbers = 30
```

The algorithm is converted to instructions in C language. The sequence of instructions to solve a problem is called program.

/*---------- Program to calculate area of a circle ---------- */

```
 #include <stdio.h>
#include <conio.h>
#define PI 3.1415
void main()
  {
        float radius, area;
      clrscr();
        printf("Enter Radius : ");
        scanf("%f", &radius);
        area = PI * radius * radius;
        printf("area of circle =%f", area);

      getch();
  }
```

OUTPUT

```
enter radius : 5
area of circle = 78.5374
```

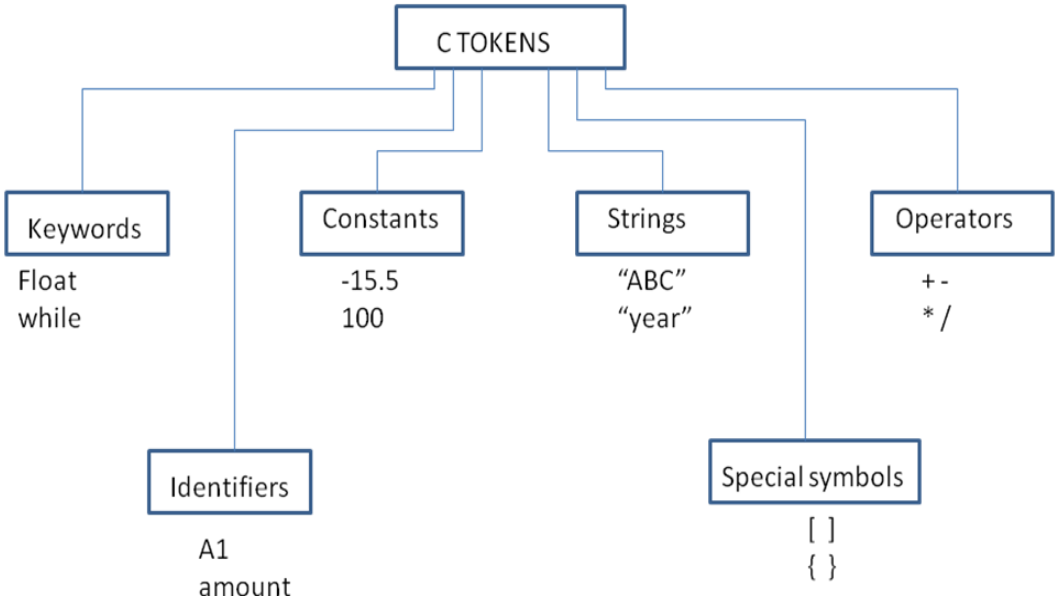
C Character Set:

The characters that can be used to form words, numbers and expressions are called as the C character set which are grouped into the following categories.

1. Letters.
2. Digits.
3. Special characters.
4. White spaces.
   The compiler ignores the white spaces unless they are a part of string constant.

C TOKENS:

In the passage of a text individual words and punctuation marks are called tokens. In C, the smallest individual units are called as c Tokens. C has Six types of tokens that are used to write the c programs.



KEYWORDS:

Every C word is classified as either a keyword or an identifier. All keywords have fixed meanings and these meanings cannot be changed. All keywords must be written in lower case. The list of all keywords in ANSI C are listed below

ANSI C KEYWORDS

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

IDENTIFIERS:

Identifiers are names given to various program elements, such as variables, functions and arrays. Identifiers consist of letters and digits, in any order, except that the first character must be a letter. Both uppercase and lowercase are permitted. Upper and lowercase letters are not interchangeable. i.e. upper case letter is not equivalent to the corresponding lowercase letter.

The under score character (_) can also be included, and is considered to be a letter. An underscore is often used in the middle of an identifier.

Valid Identifiers:

x	y12	sum
sum	area	name
TABLE	tax_rate	

Invalid Identifiers:

4th "x"
St no

DATA TYPES: C supports several different types of data, each of which may be represented differently within the computer's memory.

int, float, double, char, void

int	integer
char	single character
float	floating-point number
double	double precision floating point number Number containing a decimal point and/or an exponent. Exponent which may be larger in magnitude.
void	Nothing. normally every function returns a value. the type of function is the type of returning value. Sometimes functions will not return any value. In such case the type of that function is void.

Qualifiers: Short
Long
Signed
Unsigned

Integer quantities can be defined as

Short int
long int or
unsigned int

Short integer may require less memory than an ordinary int. Similarly long int may require more memory but it will never be less than an ordinary int.

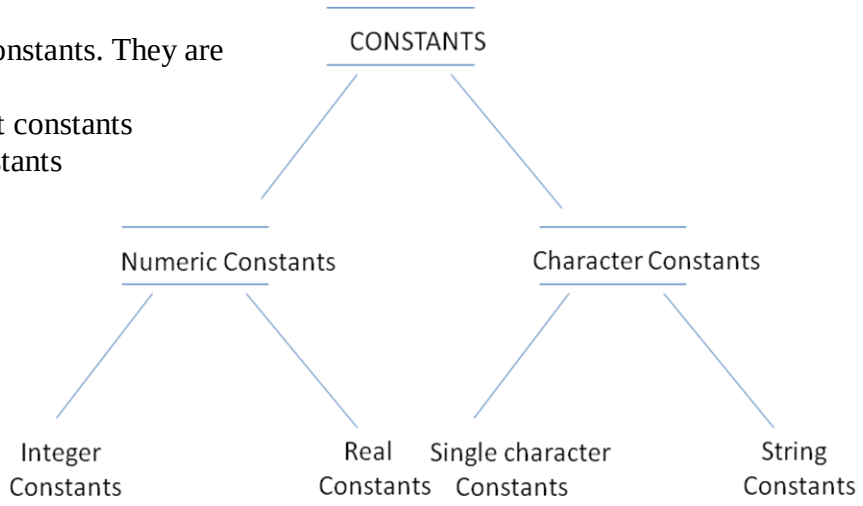
	RANGE	
Signed	-32,768	+32,767
Unsigned	0	65,535

Character type is used to represent individual characters requires 1 byte of memory.

CONSTANTS:

C has 4 basic types of constants. They are

1. Integer constants
2. Real or Floating-point constants
3. Single Character constants
4. String constants



Integer Constant:--

0 1 456 5467

Floating Constants:--

0. 1. 0.2 345.602 2E-8 0.06E-3

$3*10^5$ can be represented as follows
300000. 3E5 3E+5 3.0E+5 .3E+6 30E4

$5.026*10^{-17}$ can be represented as follows
5.026E-17

Character Constants:

A character constant is a single character enclosed in apostrophes.
'A' , 'X' , '3' , '$'

String Constants:

A string constant of any number of consecutive characters enclosed in double quotations.

"green"
"C programming language"

VARIABLE: A variable is an identifier that is used to represent some specified type of information

```
int  a, b, c;
char  d;
a = 4;
b = 3;
d =  'a';
```

All variables must be declared before they can appear in executable statements.

```
int  a,b,c;
float  root1,  root2;
char flag;
```

Expressions: An expression represents a single data item, such as a number or a character. It may also consist of some combination of such entities interconnected by one or more operators.

```
c = a + b
x = y
x <= y
x == y
++i
```

STATEMENTS: A statement causes the computer to carry out some action. There are three different classes of statements in C.

a) Expression Statement
b) Compound Statement
c) Control Statement

An expression Statement consists of an expression followed by a semicolon.

```
a = 3 ;
c = a+b;
a++;
printf("welcome to C programming ");
```

A compound statement consists of several individual statements enclosed within a pair of braces{ }.

```
{
   pi=3.1415;
   area=pi*radius*radius;
 }
```

Control Statements are used to create special program features, such as logical tests, loops and branches.

Symbolic Constants:

A symbolic Constant is a name that substitutes for a sequence of characters. The characters may represent a numeric constant, a character constant or a string constant.

```
# define name text
```

Where name represents a symbolic name, typically written in uppercase letters, and text represents the sequence of characters associated with the symbolic name. Note that text does not end with a semicolon.

```
# define TAXRATE 0.25
# define PI      3.1415
# define TRUE    1
# define FALSE   0
# define FRIEND  "rama"
```

Notice that the symbolic names are written in uppercase, to distinguish them from ordinary C identifiers.

OPERATORS

C supports rich set of operators. An operator is a symbol that tells the computer to perform certain mathematical or logical manipulations.

C operators can be classified into a number of categories. They include

1) Arithmetic operators
2) Relational operators
3) Logical operators
4) Assignment operators
5) Increment and decrement operators
6) Conditional operators
7) Cast operators
8) Bitwise operators
9) Special operators

1) Arithmetic operators : C provides all basic arithmetic operators. Operator used with only one operand is called unary operator. For example -5. The number preceded by a minus sign changes its sign.

Operator	Purpose / Meaning
+	Addition
-	Subtraction or unary minus
*	Multiplication
/	Division
%	Modulo division. Reminder after integer division

Integer division truncates any fractional part. The modulo division produces the remainder of an integer division. The modulo division operator % cannot be used on floating point data. C does not have an operator for exponentiation.

If an expression has both integer operands then the result is an integer. An arithmetic operation involving only integer operands is called **integer arithmetic**.

Example 1) Suppose a and b are integer variables such as a=10 and b=3

Expression	Value	
a+b	13	
a-b	7	
a*b	30	
a/b	3	(decimal part truncated)
a%b	1	(remainder of division)

An arithmetic operation involving only real operands is called **real arithmetic**.

Example 2) v1 v2 are floating points variables whose values are 12.5 and 2.0

Expression	Value
v1+v2	14.5
v1-v2	10.5
v1*v2	25.0
v1/v2	6.25

When one of the operands is real and the other is integer, the expression is called a mixed mode arithmetic expression. For example

15 / 10.0 = 1.5 where as
15 / 10 = 1

2)Relational Operators: We often compare two quantities, and depending on their relation, take certain decisions. For example we may compare the age of two persons, or price of two items and so on. These comparisons can be done with the help of relational operators.

Operator	Purpose/Meaning
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Example 1) To find largest number among two numbers

```
int a,b,big;

if (a>b)
      big = a;
else
      big = b;
```

Example 2) To find smallest number among two numbers

```
int a,b,small;

if (a<b)
      small = a;
else
      small = b;
```

Example 3) To check whether a given integer number is even number or odd number

```
int n, rem;

rem=n%2;

if (rem==0)
      printf("number is even number");
else
      printf("number is odd number");
```

3) Logical Operators: The logical operators are used when we want to test more than one condition and make decisions.

Operator	Purpose / Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

Ex 1) To check the result of a student. Assume that there are 3 subjects and pass mark is 40. Let m1, m2, m3 are subject marks in 3 subjects. Example using logical and operator (&&).

```
int m1, m2, m3;

if ((m1>=40)&&(m2>=40)&&(m3>=40))
        printf("pass");
else
        printf("fail");
```

Ex 2) To check the result of a student. Assume that there are 3 subjects and pass mark is 40. Let m1, m2, m3 are subject marks in 3 subjects. Example using logical or operator (||).

```
int m1, m2, m3;

if ((m1<40)||(m2<40)||(m3<40))
        printf("Fail");
else
        printf("pass");
```

Ex 3) To find largest of 3 integer numbers. Assume that a, b, c are integer numbers.

```
int a, b, c, big;

if  ((a>b)&&(a>c))
        big=a;
else
    if((b>a)&&(b>c))
            big=b;
    else
            big=c;
```

4.ASSIGNMENT OPERATORS:

identifier = expression

```
a=3;
x=y;
sum=a+b;
```

The above statements are used to assign the result of an expression to a variable. In addition to the assignment operator = C has a set of **short assignment operators**.

v op=exp;

Where v is a variable.
exp is expression
op is a C binary operator.

op = is known as Shorthand assignment operator.

v op = exp;
is equivalent to
v = v op (exp);

a = a+1;	a+=1
a = a-1	a-=1
a = a * (n+1)	a*=n+1
a = a/(n+1)	a/=n+1
a = a%b	a%=b

5. INCREMENT AND DECREMENT OPERATORS (Unary Operator)

C has two very useful operators not generally found in other languages. These are the increment and decrement Operators.

++ and - -

The operator ++ adds 1 to the operand while - - subtracts 1

++i	equivalent to	i = i + 1
i++	equivalent to	i = i + 1
--i	equivalent to	i = i – 1
i--	equivalent to	i = i – 1

The increment and decrement operators can each be utilized in the different ways, depending on placing operator before or after the operand.

++ i	pre increment
i ++	post increment
-- i	pre decrement
i--	post decrement

i=5;	**output**
printf("%d", i);	5
printf("%d", i++);	5
printf("%d", i);	6
printf("%d", ++i);	7
printf("%d", i);	7
printf("%d", i--);	7
printf("%d", i);	6
printf("%d", --i);	5
printf("%d", i);	5

Both are unary operators and take the following form.

++m	or	m++	or	m=m+1	or	m+=1;
--m	or	m--	or	m=m-1	or	m-=1

m++ and ++m mean the same thing when they form statements independently. They behave differently when they are used in expressions on the right hand side of an assignment statement.

```
m = 5;
y = ++m;
```

The value of y and m would be 6.
Suppose if we rewrite the above statements as

```
m = 5;
y = m ++;
```

The value of y would be 5 and m would be 6. A prefix operator first adds 1 to the operand and then the result is assigned to the variable on left. On the other hand, a postfix operator first assigns the value to the variable on the left and then increments the operand.

```
int a,b,c,sum;
a=5;
b=10;
sum=a+b;          5+10=15
sum=a+b++;        5+10=15
sum=a+b           5+11=16
```

6.Conditional Operator: The ?: operator

This operator is useful for making two way decisions. This operator is a combination of ? and : and takes three operands: This operator is popularly known as the conditional operator. The general form:

Conditional Expression ? expression1 : expression2 ;

The conditional expression is evaluated first. If the result is non zero exp1 is evaluated and returned as the value of conditional expression. Otherwise expression2 is evaluated and its value is returned.

Example 1) Find value of flag using conditional operator

```
if  (x<0)
   flag=0;
else
   flag=1;
```

Can be written as

```
flag= (x<0) ? 0: 1;
```

Example 2) Finding largest from two integers. we can get big by comparing two integer numbers using an if statement or using a conditional operator.

```
int  n1,n2,big;
scanf("%d%d",&n1,&n2);

big=(n1>n2) ? n1 : n2;
```

Example 3) Evaluate value of y in the following statements using conditional operator

```
y=1.5x+3          for x<=2
y=2x+5            for x>2

y=(x>2) ?  (2*x + 5) : (1.5*x + 3);
```

7.Cast Operator (Type Conversions)

The value of an expression can be converted to a different data type if desired. To do so the expression must be preceded by the name of the desired data type enclosed in parentheses.

(data type)Expression

```
int a=10;
int b=3;
a/b becomes 3
```

```
(float)a/(float)b becomes 3.33
int 10+15.5 becomes 25
```

```
int n=123;
float m;
m=(float)n;
```

the operator (float) converts n to floating point.

```
n=123
m=123.0
```

Example	Output
1. x = (int)7.5	7
2. a = (int)21.3/(int)4.5 = 21/4	5
3. b = (double) sum/n	Division is done in floating point mode
4. y = (int)a+b	Result of (a+b) is converted to int
5. z = (int)a+b	a is converted to integer and then added to b
6. p = cos((double)x)	Converts x to double before using it.

8. Bit wise Operators:

Bitwise Operators are used for manipulation of data at bit level. These operators are used for testing the bits, or shifting them right or left. Bitwise Operators may not be applied to float or double.

Operator	Meaning
&	Bitwise and
|	Bitwise or
^	Bitwise exclusive or
<<	Shift left
>>	Shift right
~	One's complement

One's complement:

a=~b;

Right shift:

```
int n, k;
n=5;
k=n>>1
```

64 >>1 …. 32
64 >> 2 …. 16
64 >> 3 …. 8
27 >> 1 … 13
49 >> 1 … 24

Left shift:

K = i << j
Value i is shifted left j No of positions
64 << 1 ———— 128
64 << 2 ———— 256

Bitwise AND: This operator is different from logical And (&&). & operates on two operands. While operating upon these two operands they are compared on a bit by bit basis. Hence both the operands must be of the same type. The second operand is often called AND MASK. The & operator operates on a pair of bits to yield a resultant bit.

```
Ex:   3 -------  0011   n1        n3= n1 & n2      n3=3&5
      5 -------  0101   n2
                 0001
```

```
Bitwise OR:-   3 -------  0011        n3=n1 | n2      n3=3|5
               5 -------  0101
                          0111
```

```
Bitwise XOR:-  3 ------  0011        n3=n1 ^ n2      n3=3^5
               5 ------  0101
                         0110
```

9. Special Operators: -

C supports some special operators of interest such as comma operator, size of operator, pointer operators (& and *) and member selection operators (. and →)

The comma permits two different expressions to appear in situations where only one expression would ordinarily be used

```
int a,b,c;
c=(a=10, b=20, a+b);
printf("%d",c);
```

first 10 is assigned to a.
then 20 is assigned to b.
10+20 is assigned to c.

sizeof () operator returns the number of bytes the operand occupies in memory. The operand may be a variable, a constant or a data type qualifier

```
sizeof (int); ---------------- 2 bytes
sizeof (float); ------------- 4 bytes
sizeof (sum); --------------- 2 bytes
sizeof (2345); ------------- 4 bytes
sizeof ('A'); --------------- 1 bytes
```

READING A CHARACTER

getchar( ): Reading a single character can be done by using the function getchar( );

varname = getchar( );

Ex: Where varname is a valid C identifier. When this statement is encountered, the computer waits until a key is pressed and then assigns this character as a value to getchar() function. Since getchar() is used on the right hand side of an assignment statement, the character is assigned to the variable.

```
char ch;
ch=getchar( );
putchar(ch);
putchar('\n');
```

PRINTING A CHARACTER

putchar (var-name)

is used to write character on the terminal. Where var-name is a character variable or character constant.

putchar ('\n'); would cause the cursor on the screen to move to the beginning of the next line.

```
char c;

c= 'a';
putchar(c);
```

Reading Multiple Values

scanf( ) :

Input data can be entered into the computer from a standard input device by means of the C library function scanf(). This function can be used to enter any combination of numerical values, single characters and strings.

```
scanf(control_string,&arg1,&arg2,……&argn);
```

Where control string refers to a string containing certain required formatting information and arg1, arg2,…. argn are arguments that represent the individual input data items.

%c single character
%d decimal integer
%e floating point value
%f floating point value
%s String
%x Hexa decimal
%o Octal integer
%u unsigned decimal integer

Each variable name must be preceded by an ampersand (&). The arguments are actually pointers, which indicate where the data items are stored in the computer memory.

```
int a;
float b;
char c;

scanf("%d",&a);
scanf("%f",&b);
scanf("%c",&c);
```

To read one integer number one floating point number one character

```
scanf("%d%f%c",&a,&b,&c);
```

To read 2 integer variables

```
int n1,n2;
scanf("%d%d",&n1,&n2);
```

To read 3 integer variables

```
int n1,n2,n3;
scanf("%d%d%d",&n1,&n2,&n3);
```

To read 2 floating point variables

```
float n1,n2;
scanf("%f%f",&n1,&n2);
```

To Read a string

```
char name[20]
scanf("%s", name);
```

Printing Multiple Values

printf( )

Output data can be written from the computer on to a standard output device using the library printf(). This function can be used to output any combination of numerical values, single characters and strings.

```
printf(control-string,arg1,arg2,……..);
```

Where control string refers to a string that contains formatting information, and arg1, arg2 are arguments that represent individual output data items.

%c single character
%d decimal integer
%e floating point value
%f floating point value
%s string
%x Hexa decimal
%o Octal

Ex 1) ----------

```
int x;
    x=20;                          output
    printf (“%d”,x);               20
```

%d is control character for an integer. In the place of %d value of integer variable from the variables list will be displayed.

```
printf (“x=%d”,x);                 x=20
printf(“value of x=%d”,x)          value of x=20
```

Ex 2)----------

```
int a,b;
a=10;
b=20;
printf(“%d  %d”,a,b);              10  20
printf(“a=%d  b=%d”,a,b);          a=10  b=20
```

as there are two %d control characters which represents two integer variables. Values of 2 integer variables from the list will be displayed in the place of %d. except control characters all other characters will be displayed as it is.

Ex 3)----------OUTPUT----------

printf(“Malineni”); Malineni

whatever the characters that are placed within double quotations will be displayed as it is

Ex 4)----------OUTPUT----------

```
printf(“Malineni”);
printf(“Perumallu”);
printf(“Educational”);
printf(“Society”);
```

MalineniPerumalluEducationalSociety

whatever the characters that are placed within double quotations will be displayed as it is. All the strings are displayed in the same line even though they are placed in different printf() statements.

Ex 5)---------- OUTPUT----------

```
printf(“Malineni\n”);
printf(“Perumallu\n”);
printf(“Educational\n”);
printf(“Society\n”);
```

Malineni
Perumallu
Educational
Society

whatever the characters that are placed within double quotations will be displayed as it is. All the strings are displayed in different lines because the new line control character \n is placed as last character in the string. \n is a new line control character which directs the compiler to go to start of new line.

WHITESPACE CHARCATERS

There is no particular symbol for whitespace. Whitespace characters are customarily:

‘ ‘ space
'\t' horizontal tab
'\n' newline
'\v' vertical tab
'\f' feed
'\r' carriage return
‘\b’ escape sequence moves cursor one position back.

```
char name1[10]=”sachin”;
char name2[10]=”ram”;
printf(“%s%s”,name1,name2);
printf(“%s\b%s”,name1,name2);
printf(“%s\b\b%s”,name1,name2);
printf(“%s\b\b\b%s”,name1,name2);
```

output
sachinram
sachiram
sachram
sacram

```
/*----------------------Floating-point format specifier : %f, %e----------*/
#include <stdio.h>
#include <conio.h>
int main()
{
   float a = 12.67;
   clrscr();
   printf("%f\n", a);
   printf("%e\n", a);
   return 0;
}
```

Output:
12.670000
1.267000e+01

```
/*------------------  Unsigned Octal number for integer : %o  ------------------------------*/
#include <stdio.h>
#include <conio.h>
void main()
{
   int a = 67;

   clrscr();
   printf("%o\n", a);
   getch();
}
```

Output:
103

```
/*----------------------Unsigned Hexadecimal for integer : %x   --------------------*/
#include <stdio.h>
#include <conio.h>
void main()
{
   int a = 15;

   clrscr();
   printf("%x\n", a);

   getch();
}
```

Output:
f

```
/*----------------------------------Double floating-point number : %lf --------*/
#include <stdio.h>
#include <conio.h>
int main()
{
    double a = 0.0;

    clrscr();
    scanf("%lf", &a);                    // input is 45.65
    printf("%lf\n", a);

    return 0;
}
```

Output:
45.650000

```
#include <stdio.h>
#include <conio.h>
void main()
{
      clrscr();
      printf("Malineni");
      printf("Perumallu");
      printf("Engineering");
      printf("College");
      printf("Guntur");
      getch();
}
```

OUTPUT

MalineniPerumalluEngineeringCollegeGuntur

```
#include <stdio.h>
#include <conio.h>
void main()
{
      clrscr();
      printf("Malineni\n");
      printf("Perumallu\n");
      printf("Engineering\n");
      printf("College\n");
      printf("Guntur\n");

      getch();
}
```

OUTPUT

Malineni
Perumallu
Engineering
College
Guntur

```
#include <stdio.h>
#include <conio.h>
void main()
{

      int a;
      a=50;
      printf("%d",a);
      getch();
}
```

OUTPUT

```
50
```

```
#include <stdio.h>
#include <conio.h>
void main()
{
      int a;

      clrscr();
      a=50;
      printf("value of a = %d",a);
      getch();
}
```

OUTPUT

```
value of a = 50
```

```
#include <stdio.h>
#include <conio.h>
void main()
{
      int a,b;

      clrscr();
      a=50;
      b=75;
      printf("%d %d\n",a,b);
      printf("a=%d b=%d\n",a,b);
      printf("value of a=%d value of b=%d\n",a,b);

      getch();
}
```

OUTPUT

```
50 75
a=50 b=75
value of a=50 value of b=75
```

```
#include <stdio.h>
#include <conio.h>
void main()
{
      float a,b;
      a=50.55;
      b=75.23;

      clrscr();
      printf("%f %f\n",a,b);
      printf("a=%f b=%f\n",a,b);
      printf("value of a=%f value of b=%f\n",a,b);

      getch();
}
```

OUTPUT

```
50.55 75.23
a=50.55 b=75.23
value of a=50.55 value of b=75.23
```

```
#include <stdio.h>
#include <conio.h>
void main()
{
      int n;

      clrscr();
      printf("enter value of n=");
      scanf("%d",&n);
      printf("value of n=%d\n",n);

      getch();
}
```

OUTPUT

```
enter value of n= 55
value of n=55
```

```
#include <stdio.h>
#include <conio.h>
void main()
{
      int n1;
      float n2;

      clrscr();
      printf("enter value of n1=");
      scanf("%d",&n1);

      printf("enter value of n2=");
      scanf("%f",&n2);

      printf("value of n1=%d\n",n1);
      printf("value of n2=%f\n",n2);

      getch();
}
```

OUTPUT

```
enter value of n1= 55
enter value of n2= 75.50

value of n1=55
value of n2=75.50
```

LIBRARY FUNCTIONS

Assume i=integer; d=double; c=character

pow() is a library function for exponential

abs(i) returns positive value of the given integer

ceil(d) returns the nearest integer greater than the number

cos(d) returns cosine of the given value

exp(d) returns e to the power d

fabs(d) returns positive value of the given integer

floor(d) returns the nearest integer less than the number

log(d) returns the logarithm of the given value

tolower(c) returns the lowercase character of the given character

toupper(c) returns the upper case character of the given character

toascii(c) returns the ASCII value of the given character

sqrt(d) returns the square root of the given number

/*---------- Program to find a to the power n ----------*/

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a,n,res;

  clrscr();
  printf("Enter value of a : ");
  scanf("%d",&a);
  printf("Enter value of n : ");
  scanf("%d",&n);
  res = pow(a,n);

  printf("%d to the power %d is %d\n",a,n,res);

  getch();
}
```

OUTPUT

```
Enter value of a : 2
Enter value of n : 3
2 to the power 3 is 8

Enter value of a : 2
Enter value of n : 6
2 to the power 6 is 64
```

```
/*----------------------Program to find square root of a given integer ------------*/
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{
  int n;
  int res;

  clrscr();
  printf("Enter a number");
  scanf("%d",&n);

  res=sqrt(n);
  printf("square root of %d is %d\n",n,res);

  getch();
}
```

OUTPUT

```
Enter a number  4
Square root of 4 is 2
```

```
/*---------------------- Program to find ASCII value of a given character ----------------------*/
#include <stdio.h>
#include <conio.h>
void main()
{
  char ch;
  int res;

  clrscr();
  printf("Enter a character : ");
  scanf("%c",&ch);

  res = toascii(ch);
  printf("The ASCII value of %c is %d\n",ch,res);

  getch();
}
```

OUTPUT

```
Enter a character : a
The ASCII value of a is 97

Enter a character : d
The ASCII value of d is 100

Enter a character : A
The ASCII value of A is 65
```

PRECEDENCE OF OPERATORS:

While executing an arithmetic statement, which has two or more operators, we may have some problems about how exactly does it get executed

For example does the expression

a+b*c correspond to (a+b) * c or a + (b*c)

The order of priority in which the operations are performed in an expression is called precedence.

An arithmetic expression with out parenthesis will be evaluated from left to right. using the rules of precedence of operators

High priority	*	/	%
Low priority	+	-	

The basic evaluation procedure includes two passes from left to right through the expression. During the first pass the high priority operators (if any) are applied as they are encountered. During the second pass the low priority operators (if any) are applied as they are encountered.

```
float  a,b,c,x,y,z;
a=9;
b=12;
c=3;
x = a-b / 3+c * 2 – 1;
y = a-b / (3+c) * (2-1);
z = a-(b/ (3+c) * 2) –1;
printf (“x = %f\n”,x);
printf(“y = %f\n”,y);
printf(“z = %f\n”,z);
……….
```

Whenever parenthesis are used the expressions within parenthesis assume highest priority. If two or more sets of parenthesis appear one after the another as 9-12/(3+3)*(2-1). The expression contained in the left most set is evaluated first and the right most in the last.

```
9-12/(3+3)*(2-1)
9-12/6*(2-1)
9-12/6*1
9-2*1
9-2
7.
```

Parenthesis may be nested, and in such cases, evaluation of the expression will proceed outward from the innermost set of parenthesis.

```
9-(12/(3+3)*2)-1
9-(12/6*2)-1
9-(2*2)-1
9-4-1
4.
```

```
9-((12/3)+3*2)-1
9-(4+3*2)-1
9-(4+6)-1
9-10-1
-2.
```

PRECEDENCE AND ASSOCIATIVITY

Each operator in C has precedence associated with it. This precedence is used to determine how an expression involving more than one operator is evaluated. The operators at the higher level of precedence are evaluated first. The operators of same precedence are evaluated either from left to right or from right to left depending on the level. This is known as associative property of an operator.

Consider the following conditional statement

If (x ==10+15 && y<10)

Addition operator has higher priority than logical operator && and relational operators = = and <. Therefore the addition of 10 and 15 is executed first. This is equal to

if(x = = 25 && y<10)

The next step is to determine whether x is equal to 25 and y is less than 10.
Assume x as 20 and y as 5 then

x = = 25 is false
y<10 is true.

The operator < has higher priority compared to = =, y<10 is tested first and then x = =25 is tested. Similarly we get if (false && true) because one of the condition is false the compare condition is false.

SIMPLE PROGRAMS

```
/* ------------------------------------------------------------
        Program to add two numbers
------------------------------------------------------------*/
#include <stdio.h>
#include <conio.h>
void main()
{
  int a,b,sum;

  clrscr();
  printf("enter a and b ");
  scanf("%d%d",&a,&b);
  sum = a + b;
  printf("sum of two numbers = %d\n",sum);
  getch();
}
```

OUTPUT

enter a and b 10 20
sum of two numbers = 30

```
/* ------------------------------------------------------------------------
        Program to multiply two numbers
------------------------------------------------------------------------*/
#include <stdio.h>
#include <conio.h>
void main()
{
  int a,b,product;

  clrscr();
  printf("enter a and b ");
  scanf("%d%d",&a,&b);
  product=a*b;
  printf("product of two numbers = %d\n",product);
  getch();
}
```

OUTPUT

enter a and b 10 20
product of two numbers = 200

```
/* ------------------------------------------------------
     Program to find quotient and remainder
------------------------------------------------------------*/
```

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a,b,quot,rem;

  clrscr();
  printf("enter a and b ");
  scanf("%d%d",&a,&b);
  quot=a/b;
  rem=a%b;
  printf("quotient = %d\n",quot);
  printf("remainder = %d\n",rem);
  getch();
}
```

OUTPUT

```
enter a and b 25 3
quotient = 8
remainder = 1
```

```
/* ------------------------------------------------------------------------------------------
     Temperature conversion Centigrade to Fahrenheat    C=(F-32)/1.8
-------------------------------------------------------------------------------------------*/
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
   float c,f;

   clrscr();
   printf("enter temperature in centigrade ");
   scanf("%f",&c);
   f = c * 1.8 +32.0;
   printf("Fahrenheat temperature =%f\n",f);
   getch();
}
```

OUTPUT

```
enter temperature in centigrade 45
Fahrenheat temperature 113.00


enter temperature in centigrade 70
Fahrenheat temperature 158.00
```

```
/* ---------------------------------------
      Program to find Area of a Circle
 ---------------------------------------*/
#include <stdio.h>
#include <conio.h>
#define PI 3.1415
void main()
{
   float r,a;

   clrscr();
   printf("enter radius ");
   scanf("%f",&r);
   a = PI * r * r;
   printf("Area of circle= %f\n",a);
   getch();
}
```

OUTPUT

enter radius 5
Area of circle 78.537

enter radius 8
Area of circle 201.056

enter radius 3
Area of circle 28.273

Write a C program to find area of a triangle if lengths of all 3 sides are given

$$s = \frac{a+b+c}{2}$$

$$area = \sqrt{s*(s-a)*(s-b)*(s-c)}$$

```
/*-------------------------------------------------------
        Program to find Area of a Triangle
 -------------------------------------------------------*/

 #include <stdio.h>
 #include <conio.h>
 #include <math.h>
 void main()
 {
    float a,b,c,s,area;

    clrscr();
    printf("enter the three sides of a triangle  ");
    scanf("%f%f%f",&a,&b,&c);
    s=(a+b+c)/2;
    area=sqrt(s*(s-a)*(s-b)*(s-c));
    printf("Area of a Triangle = %f\n",area);

   getch();
 }
```

OUTPUT

```
enter the three sides of a triangle  2 3 4
Area of a triangle = 2.904

enter the three sides of a triangle  3 5 7
Area of a triangle = 6.495
```

```
/*-------------------------------------------------------------
      Program to process student information
 -------------------------------------------------------------*/
#include <stdio.h>
#include <conio.h>
void main()
{
  int stno,m1,m2,m3,tot;
  float avg;

  clrscr();
  printf("enter student number ");
  scanf("%d",&stno);
  printf("Enter marks in 3 subjects ");
  scanf("%d%d%d",&m1,&m2,&m3);
  tot=m1+m2+m3;
  avg=(float)tot/3.0;

  printf("student number =%d \n",stno);
  printf("m1=%d  m2=%d  m3=%d\n",m1,m2,m3);
  printf("Total marks =%d\n",tot);
  printf("Average Marks =%f\n",avg);
  if ((m1<40)||(m2<40)||(m3<40))
     printf("Result=Fail\n");
  else
     printf("Result=Pass\n");
  getch();
}
```

OUTPUT

```
enter student no 556
enter marks in 3 subjects 45 56 75
student number = 556
m1=  45   m2= 56   m3 = 75
Total Marks= 176
Average Marks = 58.66
Result = Pass


enter student no 528
enter marks in 3 subjects 67  23 76
student number = 528
m1=  67   m2= 23   m3 = 76
Total Marks= 166
Average Marks = 55.33
Result = Fail
```

```
/* ------------------------------------------------------------
          Program to find electricity bill
 ------------------------------------------------------------*/
#include <stdio.h>
#include <conio.h>
void main()
{
   int sno,pmr,cmr,units;
   float unitcost,amount;

   clrscr();
   printf("Enter service number  ");
   scanf("%d",&sno);
   printf("Enter Previous meter reading");
   scanf("%d",&pmr);
   printf("Enter current meter reading");
   scanf("%d",&cmr);
   units = cmr-pmr;
   printf("Enter unit cost ");
   scanf("%f",&unitcost);
   amount = units * unitcost;
   printf("no of units =%d\n",units);
   printf("electricity  bill=%f\n",amount);

   getch();
}
```

OUTPUT

```
enter service number  236
Enter Previous meter reading 1675
Enter Current meter reading 2345
enter unit cost 5
no of units = 670
electricity bill = 3350.00
```

$$Area = \frac{b * h}{2}$$

```
/*-------------------------------------------------------------------------------
  Program to find Area of a Triangle when base and height are given
-------------------------------------------------------------------------------*/
    #include <stdio.h>
    #include <conio.h>
    #include <math.h>
    void main()
    {
      float b,h,area;

      clrscr();
      printf("enter the base and height of a triangle  ");
      scanf("%f%f",&b,&h);
      area=b*h/2;
      printf("Area of a Triangle = %f\n",area);

      getch();
    }
```

OUTPUT

```
enter base and height of a triangle 4 7
Area of a Triangle = 14.00

enter base and height of a triangle 5 8
Area of a Triangle = 20.00
```

```
/*-------------------------------------------------------------------------
        Program to interchange values of two variables
 -------------------------------------------------------------------------*/
 #include <stdio.h>
 #include <conio.h>
 void main()
  {
            int a,b,temp;

            clrscr();
            printf("enter two numbers ");
            scanf("%d%d",&a,&b);
            printf("Before interchange  a=%d   b=%d \n",a,b);

            temp=a;
            a=b;
            b=temp;

            printf("After interchange   a=%d   b=%d \n",a,b);

            getch();
     }
```

OUTPUT

enter two numbers 10 20
i.e. a=10 b=20
temp=10
a=20
b=10
Before interchanging a= 10   b=20
After interchanging   a= 20   b=10

```
/*-----------------------------------------------------------------------------------------------
Program to interchange values of two variables without using a temporary variable.
-----------------------------------------------------------------------------------------------*/
#include <stdio.h>
#include <conio.h>
void main()
 {
        int a,b;

        clrscr();
        printf("enter two numbers ");
        scanf("%d%d",&a,&b);
        printf("Before interchange  a=%d   b=%d \n",a,b);

        a=a+b;
        b=a-b;
        a=a-b;

       printf("After interchange   a=%d   b=%d \n",a,b);
       getch();
}
```

Ex: a = 10
b = 20

a = a + b = 10 + 20 = 30
b = a - b = 30 - 20 = 10
a = a - b = 30 - 10 = 20

Before interchange a=10 b=20
After interchange a=20 b=10

OUTPUT

```
enter two numbers  50 60
Before interchange a=50   b=60
After interchange   a=60   b=50
```

Problem) Write a program to find compound interest. Given amount p, years n and rate per year r. Compound Interest = p $(1+i)^n$ where i= r / 100.

$$Compund\ Interest = p*(1+\frac{r}{100})^n$$

```
/* ------------------------------------------------------
      Program to find compound interest
------------------------------------------------------*/
#include <stdio.h>
#include <conio.h>

#include <math.h>
void main()
{
  float principle,ci;
  int rate,n;

  clrscr();
  printf("enter principle amount : ");
  scanf("%f",&principle);
  printf("enter rate of interest : ");
  scanf("%d",&rate);
  printf("enter number of years ");
  scanf("%d",&n);

  ci = principle*pow((1+(float)rate/100.0),n);
  printf("compound interest = %f\n",ci);

  getch();
}
```

OUTPUT

```
enter principle amount : 10000
enter rate of interest : 12
enter number of years : 2
compound interest = 12544.00


enter principle amount : 25000
enter rate of interest : 14
enter number of years : 4
compound interest = 42224.0039
```

Standard header files in C

In C language, header files contain the set of predefined standard library functions. The "#include" preprocessing directive is used to include the header files with ".h" extension in the program.

Here is the table that displays some of the header files in C language,

Header File	Used for
stdio.h	Input/Output functions scanf() printf()
conio.h	Console Input/Output functions clrscr()
stdlib.h	General utility functions atoi()
math.h	Mathematics functions sqrt() pow()
string.h	String functions strlen() strcat()
ctype.h	Character handling functions
time.h	Date and time functions